UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/588,411	06/06/2000	Roger Wolff	13237-2575(MS-149368.1)	9449

27488 7590 04/29/2008
MERCHANT & GOULD (MICROSOFT)
P.O. BOX 2903
MINNEAPOLIS, MN 55402-0903

EXAMINER	
RUTLEDGE, AMELIA L	

ART UNIT	PAPER NUMBER
2176	

MAIL DATE	DELIVERY MODE
04/29/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	**Application No.** 09/588,411	**Applicant(s)** WOLFF ET AL.	
	Examiner AMELIA RUTLEDGE	**Art Unit** 2176	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *19 February 2008*.

2a)□ This action is **FINAL**. 2b)☒ This action is non-final.

3)□ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-3,7,8,10-14,16-19,21 and 24-27* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)□ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-3,7,8,10-14,16-19,21 and 24-27* is/are rejected.

7)□ Claim(s) _____ is/are objected to.

8)□ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)□ The specification is objected to by the Examiner.

10)□ The drawing(s) filed on _____ is/are: a)□ accepted or b)□ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)□ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)□ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)□ All b)□ Some * c)□ None of:

1.□ Certified copies of the priority documents have been received.

2.□ Certified copies of the priority documents have been received in Application No. _____.

3.□ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) □ Notice of References Cited (PTO-892)
2) □ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date *04/09/2008; 03/24/2008; 02/20/2008; 01/31/2008; 01/07/2008*.

4) □ Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ .
5) □ Notice of Informal Patent Application
6) □ Other: _____.

DETAILED ACTION

1. This action is responsive to communications: Amendment, entered 02/19/2008; RCE, filed 02/19/2008; and Information Disclosure Statements filed 04/09/2008; 03/24/2008; 02/20/2008; 01/31/2008; 01/07/2008.

2. Claims 1-3, 7, 8, 10-14, 16-19, 21, and 24-27 are pending in the case. Claims 1, 10, 19, and 27 are independent claims.

Continued Examination Under 37 CFR 1.114

A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 12/19/2007 has been entered.

Information Disclosure Statements

The information disclosure statements filed 04/09/2008; 03/24/2008; 02/20/2008; 01/31/2008; 01/07/2008 list various official communications including USPTO Office Actions, International Search Reports, and International Written Opinions. The Information Disclosure Statements also list the references which were cited in the official communications.

The prior art references listed in the IDS will be considered, however, the official communications have been lined through and will not be listed on the face of any patent issued. Official communications are not published prior art and should not be listed on the information disclosure statement, however it is proper to list the references which were cited in the official communications.

The replacement copies of Information Disclosure Statements filed with the Amendment of 12/19/2007 have been received and will be considered with the next Office Action.

Claim Rejections - 35 USC § 103

1. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims 1-3, 7, 8, 10-14, 16-19, 21, and 24-27 are rejected under 35 U.S.C. 103(a) as being unpatentable over Beauregard et al. ("Beauregard"), U.S. Patent No. 5,974,413 filed July 1997, in view of Storisteanu et al. ("Storisteanu"), U.S. Patent No. 6,976,209 B1, filed April 1999, issued December 2005.

Regarding independent claim 1, Beauregard teaches receiving a string of text in a recognizer after the entire string of text has been entered in the electronic document library in fig. 7, col. 5 lines 12-56, and col. 36 line 63 – col. 37 line 7. Beauregard teaches transmitting a string of text to a plurality of recognizer software modules in fig. 4-7 and col. 36 line 63 – col. 37 line 7. Beauregard does not explicitly teach, but suggests, the use of plug-ins, because Beauregard teaches that a user may purchase and download additional ActiveWords applications via a website, and further teaches installing and registering the downloaded application with the disclosed invention (col. 50, l. 9-68), consistent with the use of plug-ins which was common at the time of the invention. Further, Beauregard teaches that the additional downloaded applications contain recognizer libraries and Beauregard teaches the use of DLLs as agents (col. 50, l. 35-38; col. 35, l. 5-52).

Claim 1 recites *automatically receiving the string of text in a recognizer dynamic-link library after the entire string of text has been entered in the electronic document, wherein receiving the string of text comprises maintaining a job queue, the job queue storing the string of text before transmitting the string of text to a plurality of recognizer plug-ins; determining if the string of text has been edited; when the string of text has been edited, deleting the edit string of text from the queue; when the string of text has not been edited, transmitting the string of text from the job queue to the plurality of recognizer plug-ins during an idle time.* Beauregard teaches a state table which is a storage file of fixed length storing data that has been entered by a user; compare to a job queue storing text strings (col. 32, l. 5-25), Beauregard teaches a database for archiving entered and edited text (col. 32, l. 27-55). However, Beauregard does not explicitly teach storing a string of text in a job queue in a recognizer DLL. Storisteanu discloses an activemark mechanism for a live parsing editor which allows labels in text to be referenced to any editor command, macro, or external tool activated by the editor (abstract; col. 1, l. 35-col. 2, l. 64). Storisteanu discloses storing text from an editing program in a DLL after it has been entered in an electronic document (col. 22, l. 1-65; col. 21. l. 36-67), and storing semantically labeled text elements in lists, determining whether the text has been edited, and when it has been edited, deleting it from the list (col. 17, l. 1-col. 18, l. 53). Also see col. 14, l. 50-col. 16, l. 64. Storisteanu also teaches transmitting the string of text from the job queue to the plug-ins during an idle time, because Storisteanu discloses invocation of system commands through a command shell and the use of a JVM (col. 22, l. 1-65) which allocated sending commands during system idle times, such as after the editing of text.

Beauregard teaches in each of the plurality of recognizer software modules, annotating the string of text to determine a plurality of labels, wherein the plurality of labels is determined based at least on the context of the string of text in the electronic document in fig. 7, col. 5 lines 12-56, and col. 25 line 11 – col. 26 line 29. Beauregard teaches transmitting the plurality of labels from the recognizer modules to the recognizer dynamic-link library and transmitting the plurality of labels to the application program module in fig. 4-7, col. 5 lines 12-56, and col. 36 line 63 – col. 37 line 7; also see col. 35, l. 5-34 where Beauregard discloses that agents are dynamic link libraries (DLLs), especially l. 13-16. Beauregard teaches automatically receiving the string of text in a recognizer agent, i.e., DLL, after the string has been entered in the electronic document, since Beauregard teaches that the archiving agent captures all text input during a session (Col. 32, l. 28-55).

While Beauregard does not explicitly teach compiling the labels into a plurality of semantic categories at the recognizer DLL, and transmitting the semantic categories to the application program module such that each label is associated with the string of text, Storisteanu discloses storing text from an editing program in a DLL after it has been entered in an electronic document (col. 22, l. 1-65), and storing semantically labeled text elements in lists, determining whether the text has been edited, and when it has been edited, deleting it from the list (col. 17, l. 1-col. 18, l. 53). Also see col. 14, l. 50-col. 16, l. 64.

While Beauregard does not explicitly teach *embedding the plurality of semantic categories in the electronic document*, Storisteanu teaches an editor which may be programmed to encode activemarks set up during the edit session as hard-coded tags in the document file (col. 2, l. 18-26), i.e., embedding the semantic categories in the electronic document. While Storisteanu discloses that a feature of the activemark system is that no change is needed in the processed source file, because all the functionality can be handled by the live parser manipulating the document, Storisteanu specifically discloses that the parser and editor tools can also add functionality-equivalent tags to the saved source document (col. 10, l. 20-31), i.e. *embedding the plurality of semantic categories in the electronic document*.

Both Beauregard and Storisteanu are directed toward the semantic labeling of text in electronic document editing systems. It would have been obvious to one of ordinary skill in the art at the time of the invention to apply the editor API and DLL interfaces disclosed by Storisteanu to the semantic user interface disclosed by

Beauregard, since both applications utilized DLLs and Beauregard was designed to be extended with additional applications such as that disclosed by Storisteanu, so that Beauregard would have the benefit of the editing software and commands disclosed by Storisteanu.

Regarding dependent claim 2, Beauregard teaches synchronizing the labels received from the recognizer module before transmitting the plurality of labels to the application program module in col. 42 line 27 – col. 43 line 21. The labels are synchronized in order to be presented simultaneously to the user in a menu by the recognized word.

Regarding dependent claim 3, Beauregard teaches receiving the labels in an action library in fig. 7 and col. 5 lines 12-56. Beauregard teaches displaying a menu displaying a plurality of actions based on a label in fig. 9. Beauregard does not teach using action plug-in software. However, the disclosure of plug-ins is inherent in Beauregard, because Beauregard teaches that a user may purchase and download additional ActiveWords applications via a website, and further teaches installing and registering the downloaded application with the disclosed invention (col.. 50, l. 9-68), consistent with the use of plug-ins which was common at the time of the invention. Further, Beauregard teaches that the additional downloaded applications contains recognizer libraries (col. 50, l. 35-38).

Regarding dependent claim 7, Beauregard teaches causing the application program module to fire an event within an object model of the application program module and causing a piece of code associated with the event to be executed when at least one of the labels is determined in fig. 7, 9, and col. 5 lines 12-56.

Regarding dependent claim 8, Beauregard teaches determining the language of the text string based on the user profile and selecting different methods based on language precedence, and turning applications on and off based on language (Col. 25, l. 10-Col. 26, l. 19).

Regarding independent claim 10, Beauregard teaches determining whether an entered string of text matches one of a plurality of stored strings and determining an action if the string is matched in fig. 7, col. 5 lines 12-56, and col. 36 line 63 – col. 37 line 7. Beauregard teaches determining a label associated with the matched stored string, wherein the label is determined based at least on the context of the string of text in the electronic document in fig. 7, col. 5 lines 12-56, and col. 25 line 11 – col. 26 line 29. Beauregard teaches transmitting the plurality of labels from the recognizer modules to the recognizer dynamic-link library and transmitting the plurality of labels to the application program module in fig. 4-7, col. 5 lines 12-56, and col. 36 line 63 – col. 37 line 7; also see col. 35, l. 5-34 where Beauregard discloses that agents are dynamic link libraries (DLLs), (especially l. 13-16). Beauregard teaches automatically receiving the string of text in a recognizer agent, i.e., DLL, after the string has been entered in the electronic document, since Beauregard teaches that the archiving agent captures all text input during a session (Col. 32, l. 28-55). Beauregard further discloses annotating the text strings with labels and associating each label with the text string, since Beauregard teaches that each record in the archive contains the actual text stream and a tag, i.e., label, identifying the associated application and file, timestamp (Col. 32, l. 28-55, especially l. 46-60). Beauregard discloses automatic recording of the text string into the archive (Col. 34, l. 42-43).

Claim 10 recites *wherein receiving the string of text comprises maintaining a job queue, the job queue storing the string of text before transmitting the string of text to a plurality of recognizer plug-ins; determining if the string of text has been edited; when the string of text has been edited, deleting the edit string of text from the queue; when the string of text has not been edited, transmitting the string of text from the job queue to the plurality of recognizer plug-ins during an idle time.* Beauregard teaches a state table which is a storage file of fixed length storing data that has been entered by a user; compare to a job queue storing text strings (col. 32, l. 5-25), Beauregard teaches a database for archiving entered and edited text (col. 32, l. 27-55). However, Beauregard does not explicitly teach storing a string of text in a job queue in a recognizer DLL. Storisteanu discloses an activemark mechanism for a live parsing editor which allows labels in text to be referenced to any editor command, macro, or external tool activated by the editor (abstract; col. 1, l. 35-col. 2, l. 64). Storisteanu discloses storing text from an editing program in a DLL after it has been entered in an electronic document (col. 22, l. 1-65; col. 21. l. 36-67), and storing semantically labeled text elements in lists, determining whether the text has been edited, and when it has been edited, deleting it from the list (col. 17, l. 1-col. 18, l. 53). Also see col. 14, l. 50-col. 16, l. 64. Storisteanu also teaches transmitting the string of text from the job queue to the plug-ins during an idle time, because Storisteanu discloses invocation of system commands through a command shell and the use of a JVM (col. 22, l. 1-65) which allocated sending commands during system idle times, such as after the editing of text.

Beauregard teaches in each of the plurality of recognizer software modules, annotating the string of text to determine a plurality of labels, wherein the plurality of labels is determined based at least on the context of the string of text in the electronic document in fig. 7, col. 5 lines 12-56, and col. 25 line 11 – col. 26 line 29. Beauregard teaches transmitting the plurality of labels from the recognizer modules to the recognizer dynamic-link library and transmitting the plurality of labels to the application program module in fig. 4-7, col. 5 lines 12-56, and col. 36 line 63 – col. 37 line 7; also see col. 35, l. 5-34 where Beauregard discloses that agents are dynamic link libraries (DLLs), especially l. 13-16. Beauregard teaches automatically receiving the string of text in a recognizer agent, i.e., DLL, after the string has been entered in the electronic document, since Beauregard teaches that the archiving agent captures all text input during a session (Col. 32, l. 28-55).

While Beauregard does not explicitly teach compiling the labels into a plurality of semantic categories at the recognizer DLL, and transmitting the semantic categories to the application program module such that each label is associated with the string of text, Storisteanu discloses storing text from an editing program in a DLL after it has been entered in an electronic document (col. 22, l. 1-65), and storing semantically labeled text elements in lists, determining whether the text has been edited, and when it has been edited, deleting it from the list (col. 17, l. 1-col. 18, l. 53). Also see col. 14, l. 50-col. 16, l. 64.

While Beauregard does not explicitly teach *embedding the plurality of semantic categories in the electronic document*, Storisteanu teaches an editor which may be programmed to encode activemarks set up during the edit session as hard-coded tags in the document file (col. 2, l. 18-26), i.e., embedding the semantic categories in the electronic document. While Storisteanu discloses that a feature of the activemark system is that no change is needed in the processed source file, because all the functionality can be handled by the live parser manipulating the document, Storisteanu specifically discloses that the parser and editor tools can also add functionality-equivalent tags to the saved source document (col. 10, l. 20-31), i.e. *embedding the plurality of semantic categories in the electronic document*.

Both Beauregard and Storisteanu are directed toward the semantic labeling of text in electronic document editing systems. It would have been obvious to one of ordinary skill in the art at the time of the invention to apply the editor API and DLL interfaces disclosed by Storisteanu to the semantic user interface disclosed by Beauregard, since both applications utilized DLLs and Beauregard was designed to be extended with additional applications such as that disclosed by Storisteanu, so that Beauregard would have the benefit of the editing software and commands disclosed by Storisteanu.

Regarding dependent claim 11, Beauregard teaches determining a set of actions associated with a label for a string of text in fig. 7 and 9, and col. 5 lines 12-56.

Regarding dependent claim 12, Beauregard teaches displaying an indication indicating that a label has been found in fig. 9 and col. 5 lines 12-56.

Regarding dependent claim 13, Beauregard teaches determining that a user has selected a string of text and in response, displaying a plurality of actions to the user in fig. 7 and 9, col. 5 lines 12-56, and col. 36 line 63 – col. 37 line 7.

Regarding dependent claim 14, Beauregard teaches receiving an indication that one of the plurality of actions has been selected and in response to receiving an indication that one of the plurality of actions has been selected, then causing the selected action to execute in fig. 7 and 9, and col. 5 lines 12-56.

Regarding dependent claim 16, Beauregard teaches that the selected action is executed by determining whether an action library assigned to the action is available and if so, then receiving instructions from the action dynamic link library assigned to the selected action in fig. 7 and 9, and col. 5 lines 12-56.

Regarding dependent claim 17, Beauregard discloses both plug-ins and DLLs, because Beauregard teaches that a user may purchase and download additional ActiveWords applications via a website, and further teaches installing and registering the downloaded application with the disclosed invention (col.. 50, l. 9-68), consistent with the use of plug-ins which was common at the time of the invention. Further, Beauregard teaches that the additional downloaded applications contains recognizer libraries (col. 50, l. 35-38).

Regarding dependent claim 18, Beauregard discloses determining metadata associated with the string of text in the form of seven user definable subcategories which can also be automatically assigned (Col. 32, l. 61-Col. 33, l. 15).

Regarding independent claim 19, Beauregard teaches determining whether an entered string of text matches one of a plurality of stored strings and determining an action if the string is matched in fig. 7, col. 5 lines 12-56, and col. 36 line 63 – col. 37 line 7. Beauregard teaches determining a label associated with the matched stored string, wherein the label is determined based at least on the context of the string of text in the electronic document in fig. 7, col. 5 lines 12-56, and col. 25 line 11 – col. 26 line 29. Beauregard teaches transmitting the plurality of labels from the recognizer modules to the recognizer dynamic-link library and transmitting the plurality of labels to the application program module in fig. 4-7, col. 5 lines 12-56, and col. 36 line 63 – col. 37 line 7; also see col. 35, l. 5-34 where Beauregard discloses that agents are dynamic link libraries (DLLs), (especially l. 13-16). Beauregard teaches automatically receiving the string of text in a recognizer agent, i.e., DLL, after the string has been entered in the electronic document, since Beauregard teaches that the archiving agent captures all text input during a session (Col. 32, l. 28-55). Beauregard further discloses annotating the text strings with labels and associating each label with the text string, since Beauregard teaches that each record in the archive contains the actual text stream and a tag, i.e., label, identifying the associated application and file, timestamp (Col. 32, l. 28-55, especially l. 46-60). Beauregard discloses automatic recording of the text string into the archive (Col. 34, l. 42-43).

Claim 19 recites *wherein recognizer plug-in receiving the string comprises: maintaining a job queue, the job queue storing the string of text before transmitting the string of text to a plurality of recognizer plug-ins; determining if the string of text has* *been edited; when the string of text has been edited, deleting the edit string of text from the queue; when the string of text has not been edited, transmitting the string of text from the job queue to the plurality of recognizer plug-ins during an idle time.* Beauregard teaches a state table which is a storage file of fixed length storing data that has been entered by a user; compare to a job queue storing text strings (col. 32, l. 5-25), Beauregard teaches a database for archiving entered and edited text (col. 32, l. 27-55). However, Beauregard does not explicitly teach storing a string of text in a job queue in a recognizer DLL. Storisteanu discloses an activemark mechanism for a live parsing editor which allows labels in text to be referenced to any editor command, macro, or external tool activated by the editor (abstract; col. 1, l. 35-col. 2, l. 64). Storisteanu discloses storing text from an editing program in a DLL after it has been entered in an electronic document (col. 22, l. 1-65; col. 21. l. 36-67), and storing semantically labeled text elements in lists, determining whether the text has been edited, and when it has been edited, deleting it from the list (col. 17, l. 1-col. 18, l. 53). Also see col. 14, l. 50-col. 16, l. 64. Storisteanu also teaches transmitting the string of text from the job queue to the plug-ins during an idle time, because Storisteanu discloses invocation of system commands through a command shell and the use of a JVM (col. 22, l. 1-65) which allocated sending commands during system idle times, such as after the editing of text.

While Beauregard does not explicitly teach *embedding the plurality of semantic categories in the electronic document*, Storisteanu teaches an editor which may be programmed to encode activemarks set up during the edit session as hard-coded tags in the document file (col. 2, l. 18-26), i.e., embedding the semantic categories in the electronic document. While Storisteanu discloses that a feature of the activemark system is that no change is needed in the processed source file, because all the functionality can be handled by the live parser manipulating the document, Storisteanu specifically discloses that the parser and editor tools can also add functionality-equivalent tags to the saved source document (col. 10, l. 20-31), i.e. *embedding the plurality of semantic categories in the electronic document*.

Beauregard teaches in each of the plurality of recognizer software modules, annotating the string of text to determine a plurality of labels, wherein the plurality of labels is determined based at least on the context of the string of text in the electronic document in fig. 7, col. 5 lines 12-56, and col. 25 line 11 – col. 26 line 29. Beauregard teaches transmitting the plurality of labels from the recognizer modules to the recognizer dynamic-link library and transmitting the plurality of labels to the application program module in fig. 4-7, col. 5 lines 12-56, and col. 36 line 63 – col. 37 line 7; also see col. 35, l. 5-34 where Beauregard discloses that agents are dynamic link libraries (DLLs), especially l. 13-16. Beauregard teaches automatically receiving the string of text in a recognizer agent, i.e., DLL, after the string has been entered in the electronic document, since Beauregard teaches that the archiving agent captures all text input during a session (Col. 32, l. 28-55).

While Beauregard does not explicitly teach compiling the labels into a plurality of semantic categories at the recognizer DLL, and transmitting the semantic categories to the application program module such that each label is associated with the string of text, Storisteanu discloses storing text from an editing program in a DLL after it has been entered in an electronic document (col. 22, l. 1-65), and storing semantically labeled text elements in lists, determining whether the text has been edited, and when it has been edited, deleting it from the list (col. 17, l. 1-col. 18, l. 53). Also see col. 14, l. 50-col. 16, l. 64.

Both Beauregard and Storisteanu are directed toward the semantic labeling of text in electronic document editing systems. It would have been obvious to one of ordinary skill in the art at the time of the invention to apply the editor API and DLL interfaces disclosed by Storisteanu to the semantic user interface disclosed by Beauregard, since both applications utilized DLLs and Beauregard was designed to be extended with additional applications such as that disclosed by Storisteanu, so that Beauregard would have the benefit of the editing software and commands disclosed by Storisteanu.

Regarding dependent claim 21, Beauregard teaches the use of third party software in fig. 7, Beauregard discloses DLLs, and Beauregard teaches that a user may purchase and download additional ActiveWords applications via a website, and further teaches installing and registering the downloaded application with the disclosed invention (col.. 50, l. 9-68), consistent with the use of plug-ins which was common at the time of the invention.

Regarding dependent claim 24, Beauregard teaches comparing the string of text with a plurality of stored strings to determine a match and labeling the string of text with the associated stored label of the matched stored string in fig. 7, col. 5 lines 12-56, and col. 36 line 63 – col. 37 line 7.

Regarding dependent claim 25, Beauregard teaches wherein the at least one recognizer software module compares the string to a plurality of stored strings to determine whether the string matches any of the stored strings, according to semantic categories in col. 38, l. 44-65; fig. 7, col. 5 lines 12-56, and col. 36 line 63 – col. 37 l. 7.

While Beauregard does not explicitly teach *embedding the plurality of semantic categories in the electronic document*, Storisteanu teaches an editor which may be programmed to encode activemarks set up during the edit session as hard-coded tags in the document file (col. 2, l. 18-26), i.e., embedding the semantic categories in the electronic document. While Storisteanu discloses that a feature of the activemark system is that no change is needed in the processed source file, because all the functionality can be handled by the live parser manipulating the document, Storisteanu specifically discloses that the parser and editor tools can also add functionality-equivalent tags to the saved source document (col. 10, l. 20-31), i.e. *embedding the plurality of semantic categories in the electronic document*.

Both Beauregard and Storisteanu are directed toward the semantic labeling of text in electronic document editing systems. It would have been obvious to one of ordinary skill in the art at the time of the invention to apply the editor API and DLL interfaces disclosed by Storisteanu to the semantic user interface disclosed by Beauregard, since both applications utilized DLLs and Beauregard was designed to be extended with additional applications such as that disclosed by Storisteanu, so that

Beauregard would have the benefit of the editing software and commands disclosed by Storisteanu.

Regarding dependent claim 26, Beauregard teaches wherein the label is associated with a matched stored string in fig. 7, col. 5 lines 12-56, and col. 36 line 63 – col. 37 line 7.

Regarding independent claim 27, Beauregard teaches receiving a string of text in a recognizer after the entire string of text has been entered in the electronic document library in fig. 7, col. 5 lines 12-56, and col. 36 line 63 – col. 37 line 7. Beauregard teaches transmitting a string of text to a plurality of recognizer software modules in fig. 4-7 and col. 36 line 63 – col. 37 line 7. Beauregard teaches that the additional downloaded applications contains recognizer libraries (col. 50, l. 35-38). Beauregard does not explicitly teach, but suggests, the use of plug-ins, because Beauregard teaches that a user may purchase and download additional ActiveWords applications via a website, and further teaches installing and registering the downloaded application with the disclosed invention (col.. 50, l. 9-68), consistent with the use of plug-ins which was common at the time of the invention.

Claim 27 recites *wherein receiving the string of text comprises maintaining a job queue, the job queue storing the string of text before transmitting the string of text to a plurality of recognizer plug-ins; determining if the string of text has been edited; when the string of text has been edited, deleting the edit string of text from the queue; when the string of text has not been edited, transmitting the string of text from the job queue to the plurality of recognizer plug-ins during an idle time.* Beauregard teaches a state table which is a storage file of fixed length storing data that has been entered by a user; compare to a job queue storing text strings (col. 32, l. 5-25), Beauregard teaches a database for archiving entered and edited text (col. 32, l. 27-55). However, Beauregard does not explicitly teach storing a string of text in a job queue in a recognizer DLL. Storisteanu discloses an activemark mechanism for a live parsing editor which allows labels in text to be referenced to any editor command, macro, or external tool activated by the editor (abstract; col. 1, l. 35-col. 2, l. 64). Storisteanu discloses storing text from an editing program in a DLL after it has been entered in an electronic document (col. 22, l. 1-65; col. 21. l. 36-67), and storing semantically labeled text elements in lists, determining whether the text has been edited, and when it has been edited, deleting it from the list (col. 17, l. 1-col. 18, l. 53). Also see col. 14, l. 50-col. 16, l. 64. Storisteanu also teaches transmitting the string of text from the job queue to the plug-ins during an idle time, because Storisteanu discloses invocation of system commands through a command shell and the use of a JVM (col. 22, l. 1-65) which allocated sending commands during system idle times, such as after the editing of text.

Beauregard teaches in each of the plurality of recognizer software modules, annotating the string of text to determine a plurality of labels, wherein the plurality of labels is determined based at least on the context of the string of text in the electronic document in fig. 7, col. 5 lines 12-56, and col. 25 line 11 – col. 26 line 29. Beauregard teaches transmitting the plurality of labels from the recognizer modules to the recognizer dynamic-link library and transmitting the plurality of labels to the application program module in fig. 4-7, col. 5 lines 12-56, and col. 36 line 63 – col. 37 line 7; also see col. 35, l. 5-34 where Beauregard discloses that agents are dynamic link libraries (DLLs), especially l. 13-16. Beauregard teaches automatically receiving the string of text in a recognizer agent, i.e., DLL, after the string has been entered in the electronic document, since Beauregard teaches that the archiving agent captures all text input during a session (Col. 32, l. 28-55).

While Beauregard does not explicitly teach *embedding the plurality of semantic categories in the electronic document*, Storisteanu teaches an editor which may be programmed to encode activemarks set up during the edit session as hard-coded tags in the document file (col. 2, l. 18-26), i.e., embedding the semantic categories in the electronic document. While Storisteanu discloses that a feature of the activemark system is that no change is needed in the processed source file, because all the functionality can be handled by the live parser manipulating the document, Storisteanu specifically discloses that the parser and editor tools can also add functionality-equivalent tags to the saved source document (col. 10, l. 20-31), i.e. *embedding the plurality of semantic categories in the electronic document*.

While Beauregard does not explicitly teach compiling the labels into a plurality of semantic categories at the recognizer DLL, and transmitting the semantic categories to the application program module such that each label is associated with the string of text, Storisteanu discloses storing text from an editing program in a DLL after it has been entered in an electronic document (col. 22, l. 1-65), and storing semantically labeled text elements in lists, determining whether the text has been edited, and when it has been edited, deleting it from the list (col. 17, l. 1-col. 18, l. 53). Also see col. 14, l. 50-col. 16, l. 64.

Both Beauregard and Storisteanu are directed toward the semantic labeling of text in electronic document editing systems. It would have been obvious to one of ordinary skill in the art at the time of the invention to apply the editor API and DLL interfaces disclosed by Storisteanu to the semantic user interface disclosed by Beauregard, since both applications utilized DLLs and Beauregard was designed to be extended with additional applications such as that disclosed by Storisteanu, so that Beauregard would have the benefit of the editing software and commands disclosed by Storisteanu.

Response to Arguments

Applicant's arguments filed 12/19/2007 have been fully considered but they are not persuasive.

Applicant's arguments are addressed to the newly claimed limitation of independent claims 1, 10, 19, and 27 (see Remarks, p. 11-13), *embedding the plurality of semantic categories in the electronic document* (claim 1). The newly claimed limitation is disclosed by Storisteanu. Applicant's specification at p. 16-17 discloses the method of embedding semantic categories as tags in a document.

While Beauregard does not explicitly teach *embedding the plurality of semantic categories in the electronic document*, Storisteanu teaches an editor which may be programmed to encode activemarks set up during the edit session as hard-coded tags in the document file (col. 2, l. 18-26), i.e., embedding the semantic categories in the electronic document. While Storisteanu discloses that a feature of the activemark system is that no change is needed in the processed source file, because all the functionality can be handled by the live parser manipulating the document, Storisteanu specifically discloses that the parser and editor tools can also add functionality-equivalent tags to the saved source document (col. 10, l. 20-31), i.e. *embedding the plurality of semantic categories in the electronic document*.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to AMELIA RUTLEDGE whose telephone number is (571)272-7508. The examiner can normally be reached on Monday - Friday 9:30 - 6:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Doug Hutton can be reached on 571-272-4137. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Amelia Rutledge/
Examiner, Art Unit 2176